一、策略评价指标

```
# -*- coding: utf-8 -*-
"""
@author: Xingbuxing
date: 2017年05月14日
本段程序用于演示策略评价指标
"""
import pandas as pd  # 导入pandas，我们一般为pandas去一个别名叫做pd
import program.config
pd.set_option('expand_frame_repr', False)  # 当列太多时不换行


# ===导入数据
select_stock = pd.read_hdf(program.config.output_data_path + '/市值选股结果.h5', 'select_stock')
equity = pd.read_hdf(program.config.output_data_path + '/市值选股结果.h5', 'equity')

# ===对资金曲线从收益角度进行评价
# 总收益
total_return = (equity.iloc[-1]['equity_curve'] / 1) - 1

# 年华收益
# pow(x, y),计算x的y次方
# 年华收益：pow((1 + x), 年数) = 总收益
# 日化收益：pow((1 + x), 天数) = 总收益
# pow((1 + 日化收益), 365) = 年华收益
# 整理得到：年华收益 = pow(总收益, 365/天数) - 1
trading_days = (equity['交易日期'].iloc[-1] -  equity['交易日期'].iloc[0]).days + 1
annual_return = pow(total_return, 365.0/trading_days) - 1
print '年华收益（%）：', round(annual_return * 100, 2)

# ===对资金曲线从风险角度进行评价
# 使用每日收益的方差衡量风险
print equity['每日涨幅'].std()

# 使用最大回撤衡量风险
# 最大回撤：从某一个高点，到之后的某个低点，之间最大的下跌幅度。实际意义：在最最坏的情况下，会亏多少钱。
# 计算当日之前的资金曲线的最高点
equity['max2here'] = equity['equity_curve'].expanding().max()
# 计算到历史最高值到当日的跌幅，drowdwon
equity['dd2here'] = equity['equity_curve'] / equity['max2here'] - 1
# 计算最大回撤，以及最大回撤结束时间
end_date, max_draw_down = tuple(equity.sort_values(by=['dd2here']).iloc[0][['交易日期',
'dd2here']])
print '最大回撤（%）：', round(max_draw_down * 100, 2)
# 计算最大回撤开始时间
start_date = equity[equity['交易日期'] <= end_date].sort_values(by='equity_curve',
ascending=False).iloc[0]['交易日期']
print '最大回撤开始时间', start_date.date()
print '最大回撤结束时间', end_date.date()
```

```
# ===终极指标
# 如果只用一个指标来衡量策略，我最常用的就是：年化收益 / abs(最大回撤)
# 越高越好，一般这个指标大于1，说明这个策略值得进一步探索。


# ===对每次操作的评价
# 平均涨幅
print '平均涨幅(%)', round(select_stock['下月平均涨跌幅'].mean() * 100, 2)

# 胜率
print '涨幅>0比例(%)', select_stock[select_stock['下月平均涨跌幅'] > 0].shape[0] /
float(select_stock.shape[0]) * 100

# 胜率
print '跑赢同期均值比例(%)', select_stock[select_stock['下月平均涨跌幅'] > select_stock['所有股票
下月平均涨跌幅']].shape[0] / float(select_stock.shape[0]) * 100

# 最大单月涨幅，最大单月跌幅
print '最大单月涨幅(%)：', round(select_stock['下月平均涨跌幅'].max() * 100, 2)
print '最大单月跌幅(%)：', round(select_stock['下月平均涨跌幅'].min() * 100, 2)  # 单月最大下跌
30%，是否还有信心坚持策略？


# 其他指标如夏普比率，信息比率，详见：http://bbs.pinggu.org/thread-4454429-1-1.html
```

二、选股数据整理

```
# -*- coding: utf-8 -*-
"""
@author: Xingbuxing
date: 2017年05月14日
本段程序用于生成选股策略所需要的数据
"""
import pandas as pd
from program import config
from program import Functions
pd.set_option('expand_frame_repr', False)  # 当列太多时不换行


# ===读取所有股票代码的列表
stock_code_list = Functions.get_stock_code_list_in_one_dir(config.input_data_path+'/stock_data')
stock_code_list = stock_code_list[:20]

# ===循环读取并且合并
# 导入上证指数
index_data = Functions.import_sh000001_data()[['交易日期']]
# 循环读取股票数据
all_stock_data = pd.DataFrame()
for code in stock_code_list:

    print code
```

```
    # 读入数据，额外读入'总市值'这一列
    df = Functions.import_stock_data(code, other_columns=['总市值'])
    # 将股票和上证指数合并，补全停牌的日期
    df = Functions.merge_with_index_data(df, index_data)
    # 将日线数据转化为月线，并且计算'是否交易'、'最后一天涨跌幅'、'交易天数'、'市场交易天数'
    # 并且计算'每天资金曲线'，excel展示计算原理
    df = Functions.transfer_to_period_data(df, period_type='m')

    # 对数据进行整理
    # 删除上市的第一个月
    df.drop([0], axis=0, inplace=True)  # 删除第一行数据
    # 开始时间太早
    df = df[df['交易日期'] > pd.to_datetime('20051201')]
    # 计算下月每天涨幅、下月涨幅
    df['下月每天资金曲线'] = df['每天资金曲线'].shift(-1)
    del df['每天资金曲线']
    df.dropna(subset=['下月每天资金曲线'], inplace=True)
    df['下月涨跌幅'] = df['涨跌幅'].shift(-1)
    # 删除当月最后交易日不交易、涨停的股票，因为这些股票在不能买入
    df = df[df['是否交易'] == 1]
    df = df[df['最后一天涨跌幅'] <= 0.097]
    # 删除交易天数过少的月份
    df = df[df['交易天数'] / df['市场交易天数'] >= 0.8]

    # 合并数据
    all_stock_data = all_stock_data.append(df, ignore_index=True)


# 将数据存入数据库之前，先排序、reset_index
all_stock_data.sort_values(['交易日期', '股票代码'], inplace=True)
all_stock_data.reset_index(inplace=True, drop=True)
print '股票数据行数', len(all_stock_data)

# 将数据存储到hdf文件
all_stock_data.to_hdf(config.output_data_path + '/all_stock_data_sample.h5', 'all_stock_data',
mode='w')
# all_stock_data.to_hdf(config.output_data_path + '/all_stock_data.h5', 'all_stock_data',
mode='w')

# 目前我们只根据市值选股，所以数据中只有一些基本数据加上市值。
# 实际操作中，会根据很多指标进行选股，
# 需要将这些指标都存入到all_stock_data.h5中。
```

三、选股

```
# -*- coding: utf-8 -*-
"""
@author: Xingbuxing
date: 2017年05月14日
本段程序演示选股策略框架

"""
```

```
import pandas as pd
import numpy as np
from program import config
from program import Functions
import matplotlib.pyplot as plt
pd.set_option('expand_frame_repr', False)  # 当列太多时不换行


select_stock = pd.DataFrame()  # select_stock用户存储选出的股票数据

# 从hdf文件中读取数据
# stock_data = pd.read_hdf(config.output_data_path + '/all_stock_data.h5', 'all_stock_data')
stock_data = pd.read_hdf(config.output_data_path + '/all_stock_data_sample.h5',
'all_stock_data')

# 所有股票下月平均涨幅，作为比较的benchmark。也可以使用指数的涨幅作为benchmark
select_stock['所有股票下月平均涨跌幅'] = stock_data.groupby('交易日期')['下月涨跌幅'].mean()

# 根据市值对股票进行排名
stock_data['排名'] = stock_data.groupby('交易日期')['总市值'].rank()

# 选取前三的股票
stock_data = stock_data[stock_data['排名'] <= 10]
stock_data['股票代码'] += ' '

# 计算每月资金曲线
stock_data_group = stock_data.groupby('交易日期')
select_stock['股票代码'] = stock_data_group['股票代码'].sum()
select_stock['下月平均涨跌幅'] = stock_data_group['下月涨跌幅'].mean()

select_stock['下月每天平均涨跌幅'] = stock_data_group['下月每天资金曲线'].apply(lambda x:
list(pd.DataFrame([1.0] + list(np.array(list(x)).mean(axis=0))).pct_change()[0])[1:])
# x = stock_data.iloc[:3]['下月每天资金曲线']
# print x
# print list(x), len(list(x))  # 将x变成list
# print np.array(list(x))  # 矩阵化
# print np.array(list(x)).mean(axis=0)  # 求每天的资金曲线
# print list(np.array(list(x)).mean(axis=0))
# print [1] + list(np.array(list(x)).mean(axis=0))
# print pd.DataFrame([1] + list(np.array(list(x)).mean(axis=0)))
# print pd.DataFrame([1] + list(np.array(list(x)).mean(axis=0))).pct_change()[0]
# print list(pd.DataFrame([1] + list(np.array(list(x)).mean(axis=0))).pct_change()[0])
# print list(pd.DataFrame([1] + list(np.array(list(x)).mean(axis=0))).pct_change()[0])[1:]
select_stock.reset_index(inplace=True)
select_stock['资金曲线'] =  (select_stock['下月平均涨跌幅'] + 1).cumprod()

# 计算每日资金曲线
index_data = Functions.import_sh000001_data()
equity = pd.merge(left=index_data, right=select_stock[['交易日期', '股票代码']], on=['交易日期'],
                  how='left', sort=True)  # 将选股结果和大盘指数合并

equity['股票代码'] = equity['股票代码'].shift()

equity['股票代码'].fillna(method='ffill', inplace=True)
```

```
equity.dropna(subset=['股票代码'], inplace=True)

equity['每日涨幅'] = select_stock['下月每天平均涨跌幅'].sum()
# print select_stock[['交易日期', '下月每天平均涨跌幅']]
# print select_stock['下月每天平均涨跌幅'].sum()

equity['equity_curve'] = (equity['每日涨幅'] + 1).cumprod()
equity['benchmark'] = (equity['大盘涨跌幅'] + 1).cumprod()
print equity

# 存储数据
# select_stock.reset_index(inplace=True, drop=True)
# select_stock = select_stock[['交易日期', '股票代码', '下月平均涨跌幅', '所有股票下月平均涨跌幅',
'资金曲线']]
# print select_stock
# select_stock.to_hdf(config.output_data_path + '/市值选股结果.h5', 'select_stock', mode='w')
# equity.reset_index(inplace=True, drop=True)
# equity = equity[['交易日期', '股票代码', '每日涨幅', 'equity_curve', '大盘涨跌幅', 'benchmark']]
# print equity
# equity.to_hdf(config.output_data_path + '/市值选股结果.h5', 'equity', mode='a')

# 画图
# equity.set_index('交易日期', inplace=True)
# plt.plot(equity['equity_curve'])
# plt.plot(equity['benchmark'])
# plt.legend(loc='best')
# plt.show()
```